A workflow for designing stylized shading effects

Alexandre Bléron, Romain Vergne, Thomas Hurtut, Joëlle Thollot

A workflow for designing stylized shading effects

Alexandre Bléron, Romain Vergne, Thomas Hurtut, Joelle Thollot

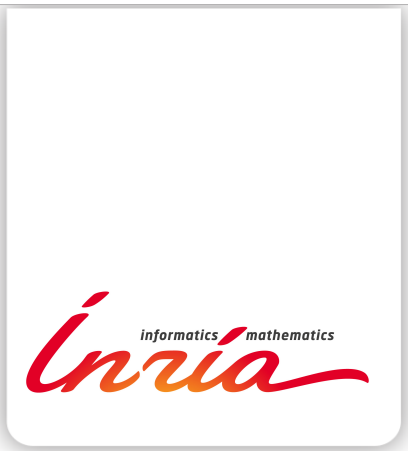
RESEARCH REPORT

N° 9225

October 2018

Project-Team Maverick

ISSN 0249-6399 ISRN INRIA/RR--9225--FR+ENG



A workflow for designing stylized shading effects

Alexandre Bléron, Romain Vergne, Thomas Hurtut, Joelle Thollot

Project-Team Maverick

Research Report n° 9225 — October 2018 — 29 pages

Abstract: In this report, we describe a workflow for designing stylized shading effects on a 3D object, targeted at technical artists. Shading design, the process of making the illumination of an object in a 3D scene match an artist vision, is usually a time-consuming task because of the complex interactions between materials, geometry, and lighting environment. Physically based methods tend to provide an intuitive and coherent workflow for artists, but they are of limited use in the context of non-photorealistic shading styles. On the other hand, existing stylized shading techniques are either too specialized or require considerable hand-tuning of unintuitive parameters to give a satisfactory result. Our contribution is to separate the design process of individual shading effects in three independent stages: control of its global behavior on the object, addition of procedural details, and colorization. Inspired by the formulation of existing shading models, we expose different shading behaviors to the artist through *parametrizations*, which have a meaningful visual interpretation. Multiple shading effects can then be composited to obtain complex dynamic appearances. The proposed workflow is fully interactive, with real-time feedback, and allows the intuitive exploration of stylized shading effects, while keeping coherence under varying viewpoints and light configurations. Furthermore, our method makes use of the *deferred shading* technique, making it easily integrable in existing rendering pipelines.

Key-words: non-photorealistic rendering, shading model, real-time rendering

RESEARCH CENTRE
GRENOBLE – RHÔNE-ALPES

Inovallée
655 avenue de l'Europe Montbonnot
38334 Saint Ismier Cedex

Un outil de conception d'effets d'illumination stylisés

Résumé : Dans ce rapport, nous décrivons un outil de création de modèles d'illumination adapté à la stylisation de scènes 3D. Contrairement aux modèles d'illumination photoréalistes, qui suivent des contraintes physiques, les modèles d'illumination stylisés répondent à des contraintes artistiques, souvent inspirées de la représentation de la lumière en illustration. Pour cela, la conception de ces modèles stylisés est souvent complexe et coûteuse en temps. De plus, ils doivent produire un résultat cohérent sous une multitude d'angles de vue et d'éclairages. Nous proposons une méthode qui facilite la création d'effets d'illumination stylisés, en décomposant le processus en trois parties indépendantes: contrôle du comportement global de l'illumination, ajout de détails procéduraux, et colorisation. Différents comportements d'illumination sont accessibles à travers des *paramétrisations*, qui ont une interprétation visuelle, et qui peuvent être combinées pour obtenir des apparences plus complexes. La méthode proposée est interactive, et permet l'exploration efficace de modèles d'illumination stylisés. La méthode est implémentée avec la technique de *deferred shading*, ce qui la rend facilement utilisable dans des pipelines de rendu existants.

Mots-clés : rendu non-photoréaliste, modèle d'illumination, rendu temps-réel

Contents

1 Introduction

When depicting objects, illustrators use light and shade in ways that deviate significantly from realism and physical rules of light propagation. Instead, it is often used to emphasize a part of the scene that the artist deems more important, or to convey a particular mood or emotion. One of the goals of *stylized shading* techniques is to translate artistic rules for the depiction of light and shade into formalized models that can be applied automatically in the context of the rendering of animated 3D scenes on a computer. This is a difficult problem because, in general, stylized lighting effects do not reflect a physical truth and as such are not easily simulated. This difficulty is exacerbated by the fact that for some of these effects, the only 2D reference available is static: there is no consensus on how a stylized light-and-shade depiction should look like under movement of the object or of the viewpoint. They must be rendered and animated by taking into account the intention of the artist, and they are not easily unified under a single rendering framework.

In 3D computer graphics, light and shade is a part of *appearance design*, which is the process of adjusting materials, surface details, and lighting to achieve a desired look for an object in a scene. As with 2D illustration, it is possible to tweak shading and lighting in 3D scenes to achieve emphasis of a particular object, abstraction or to convey mood. However, shading design

is a complex process involving multiple interconnected aspects: the appearance of an object in a 3D scene is the result of the interaction between the material, the object geometry, lighting environment, and camera viewpoint. And contrary to 2D illustration, the artist may not have control over the viewpoint of the camera or even the lighting (e.g. in interactive art applications, such as video games). Because of that, designing a shading that accurately reproduces a desired appearance and that stays visually coherent under varying viewpoints and lighting conditions is a challenge requiring both artistic and technical skills. In many industries, such as computer animation and video games, this task is often entrusted to dedicated artists.

Using physically-based models for shading alleviates some of that complexity: as their formulation and parameters are derived from physical phenomena, they ensure a coherent and plausible appearance for any geometry, under any viewpoint and lighting environment. They also have the advantage of working almost directly with captured data (e.g. captured lighting environments, or scanned materials), which greatly reduces the time spent in designing materials. However, they are by design unsuitable for reproducing any kind of non-physical lighting effect commonly used in illustration.

In contrast, stylized shading models are not physically correct or plausible. Their goals are related to depiction rather than accurate light simulation. For instance, stylized shading models are used in visualization to communicate information about the shape and material of an object in a more efficient way than realistic shading models. An example of this is Gooch shading [Goo+98], which modifies in a non-physical way the illumination term to reveal surfaces in shadow. Some models have been designed to reproduce particular artistic styles in 2D illustration: for example, the well-known and well-studied *toon shading* technique was designed to mimic styles found in comic books. However, in general, shading rules in illustration are hard to characterize, because contrary to physically based models, they are not directly linked to geometric and physical properties of the scene. Because of that, many stylized shading techniques are limited to specific styles, and usually simple dynamic behaviors.

In this report, we present a method, targeted at technical artists, for designing and exploring complex stylized shading effects by combining simple building blocks, in the form of simple shading behaviors. The proposed building blocks have intuitive visual interpretations: our intent is to make them easy to combine in order to create complex shading behaviors by progressive refinement. Going further, our intent is also to provide a practical framework for decomposing complex shading effects found in illustration, and more generally, to facilitate the exploration of the design space of stylized shading in 3D scenes.

Our main contribution is to decouple the design of individual shading effects in three independent aspects: (1) Choosing and tuning the global *shading behavior* (diffuse, specular, rim lighting, etc.) of an effect; (2) Adding details and visual complexity; (3) Colorizing the effect.

The final stylized appearance is then obtained by layering several of those shading effects. We provide interactive tools to edit each aspect, allowing the artist to precisely tune each part of the final appearance with a direct visual feedback. Our method automatically keeps a consistent result under varying viewpoints and light configurations. We show that our approach can be used to add spatially and temporally coherent details, mimicking various shading effects in a direct and flexible manner.

The structure of this report is as follows: first, in Section 2, we detail some characteristics of shading depiction in 2D illustration that motivated our approach. In Section 3, we review existing techniques for stylized lighting and shading techniques for 3D scenes. We present our technique in Sections 4 to 8, and show results in Section 9. We discuss possible future improvements and research areas in Section 10.

2 Light and shade in 2D illustration

One of the motivations for proposing such a system is the growing demand for the production of 3D content in a style inspired from traditional 2D illustrative shading styles. However, the concept of light and shade in 2D illustration is very broad: a depicted surface can be said to "catch the light" in many different ways, usually non-physical, and sometimes not consistent between different objects. An artist can play on the depiction of shading to various ends: as clues for the material of an object, to enhance specific features of a shape, to attract the focus of the spectator to some location on the image, or to set the mood of a scene. For instance, Hogarth [Hog91] proposed five different categories of light and shade in 2D illustrations with traditional media (pencil, pen-and-ink, charcoal, etc.). Each of them have different purposes: for instance, the so-called *sculptural light* is used to ensure that all details of the form of an object are revealed, regardless of whether there is an actual light shining on them.

Additionally, there are even more different rules and techniques when color is added. Custom color gradients can be used in place of photographically accurate illumination gradients for emphasis, abstraction, or as shortcuts for complex lighting effects (Figure 1) or convey the appearance of particular materials (Figure 2).

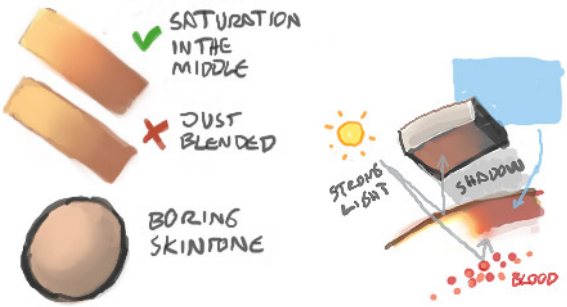


Figure 1: Illumination gradients in illustration are often modified in non-physical ways. In this example, taken from a digital painting tutorial, illumination gradients on human skin are made more appealing by artifically increasing the saturation at the transition between lit and shadowed (left). A similar technique is used to emulate the visual appearance of subsurface scattering of strong lights inside the skin (right). Source: *Basic to Advanced Color Theory and Illustration Techniques for Photoshop* `http://www.floobynooby.com/ICG/artvalues.html`

This great variety in the depiction of light and shade in both traditional and digital illustration raises the traditional question in stylized rendering: how to reproduce these appearances automatically on 3D scenes? As with other sub-domains of stylized rendering, stylized *shading* is no exception: the artistic rules for light and shade depiction are the rules of artists, and differ from traditional computer graphics or physical models in significant ways. They vary in subtle ways from artist to artist, which makes it even more difficult to propose a generalized model for stylized shading.

Adding interactivity and dynamic behavior also makes the design process more difficult, as the behavior of lighting effects must now be coherent with motion in addition to shape. For instance, some effects like metallic highlights are expected to slide on the object when moving the viewpoint, contrary to diffuse lighting which stays fixed to a surface until the light themselves are moved. Thus, illustration tricks like the one shown in Figure 2 to convey metallic appearances will not work in dynamic 3D scenes. In a way, this means that, under animation, some "important"

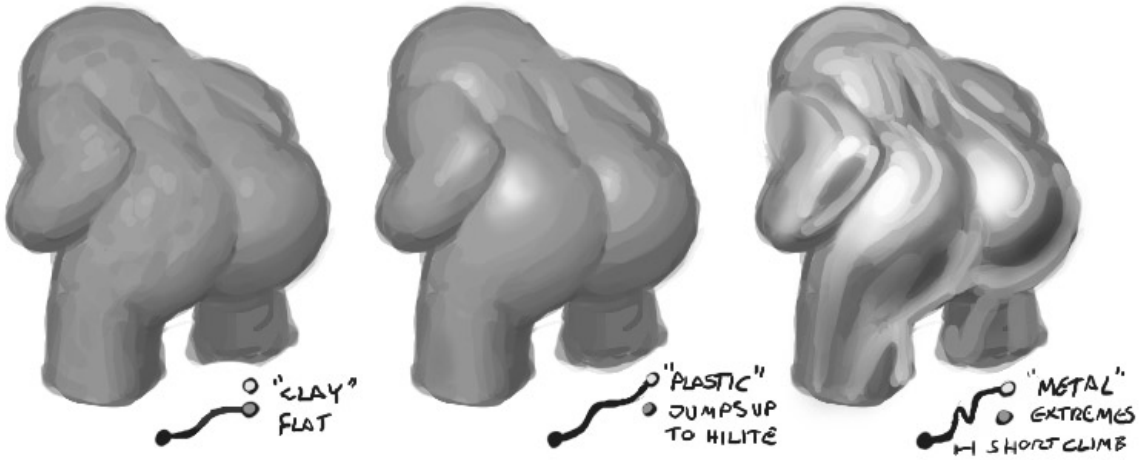


Figure 2: In the lambertian shading model, the perceived brightness of a lit surface (intensity profile) falls off linearly according to the geometry term (the cosine of the angle between the surface normal and the light direction), resulting in a smooth illumination gradient that conveys diffuse appearance (left). In illustration, simple modifications to this profile can convey dramatically different clues about the material: making the gradient "jump" when facing the light gives the impression of a specular highlight (middle), while introducing a small jump in the middle of the gradient gives off a metallic look (right). However, translating this technique to 3D is non-trivial because the appearance has to be coherent under dynamic viewpoints and lighting conditions. Source: see above.

realistic behaviors of light and shade must be kept so that the intended appearance is not broken.

While this makes stylized shading in dynamic 3D scenes more challenging, it also greatly expands the design space for artists. This observation led us to the system presented in this report, which strives to allow artists to easily explore this design space with basic shading primitives. Those primitives are not necessarily physically-based, but instead are meant to have an intuitive interpretation in terms of how they affect the final appearance.

3 Related work

In computer graphics, stylized depiction of light and shade in a scene can be achieved through several ways: by manipulating the lights themselves, or by changing the way light interacts with a surface, through specialized shading models. First, we review the techniques that manipulate the lighting environment (position, intensity, color of lights) to achieve a stylized look. Note that, with those techniques, the simulation of light can still follow physical rules. Then, we review existing literature on shading models that achieve non-photorealistic appearances.

3.1 Editing lighting environments

One form of lighting manipulation is *inverse lighting* techniques: given some artistic constraints on the final image (e.g. the position and color of an highlight, the size of a shadow cast by an object, etc.), the goal is to infer the parameters of the lights in the scene: their number,

position, color, and type. Notable work in this category include work by Pellacini et al. [Pel+07]. However, inverse lighting is usually expressed as a complex non-linear optimization problem: it thus suffers from the common issue that the optimization result may not match the intent of the artist. Several techniques propose a more direct control over the appearance by allowing the user to make non-physical edits to an existing physical result, allowing the user to move and deform reflections [Rit+09], shadows, highlights, refractions [Rit+10; Sch+13], or even the propagation of light rays [KPD10]. However, deforming existing physical effects are impractical for stylized appearances that differ significantly from realistic results.

In contrast, the approach of Okabe et al. [Oka+07] allows users to design image-based lighting environments from scratch by directly painting the desired appearance on a 3D model. Their tool supports arbitrary BRDF models. Still, these techniques cannot be extended to take into account other scene or object attributes typically used for stylization, such as surface curvature, and as such have limited flexibility for representing arbitrary stylized shadings. Lighting environments and materials can also be acquired from real objects. Providing tools that allows artists to intuitively edit captured environments after acquisition is an important area of work [Pel10; Zub+15].

All these methods are mainly used to provide artistic direction for physically-based simulation while maintaining a plausible realistic appearance: fundamentally altering the behavior and appearance of light and shade is not their primary goal. Furthermore, a wide range of shading effects found in 2D illustration cannot be reproduced solely by tweaking the lighting environment: stylization often depends on other attributes of the scene, such as surface curvature. For such effects, specialized shading models must be used.

Lit-spheres *Lit-Spheres* (also called *matcaps*) provide another direct and flexible way to specify the appearance of an object. In the system originally described by Sloan et al. [Slo+01], the appearance is represented by an image of a sphere. The target object is then shaded by environment mapping using this image. This image can be captured, reconstructed from a drawing or a photograph, or painted using digital painting tools. It can accommodate multiple shading styles, from realistic to toon-like, without the unintuitive control imposed by a BRDF. They give immediate plausible results with little to no manual adjustment required, and are now widely used for shading in 3D modeling software [1] [2]. However, they are limited to static lighting as the final appearance depends only on the camera viewpoint. Note that lit-spheres can capture not only shading but also textural details of the appearance. However, those details are subject to stretching and compression artifacts on shapes with a lot of curvature variations. Todo, Anjyo, and Yokoyama [TAY13] extended this technique by defining the lit-sphere in a light-dependent space, allowing dynamic lighting environments, and further refined it by adding brush stroke effects and highlight shape control that are not subject to deformations (Figure 3). However, the range of dynamic behaviors (i.e. how the shading reacts to position and viewpoint changes) that can be modeled with this technique is still limited.

3.2 Stylized shading models

Simple shading models derived from physical principles of light propagation are usually not suited to reproduce light and shade found in illustrations, as they do not convey information about a shape in the most effective way possible. To illustrate this, let's consider lambertian shading, used in computer graphics as a physical model of purely diffuse surfaces. In this model, a color

[1] Pixologic ZBrush Features `http://pixologic.com/zbrush/features/Materials/`

[2] Shading - Blender Manual `https://docs.blender.org/manual/nb/dev/editors/3dview/properties/shading.html`

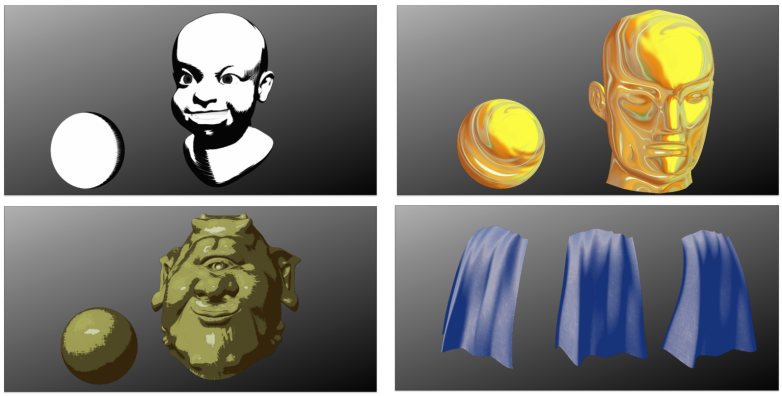


Figure 3: Examples of lit-sphere shading with the method of Todo, Anjyo, and Yokoyama [TAY13]. Images taken from the paper.

term c is multiplied by the incoming light I_{in} and modulated by the *geometric term* ($\max(0, \mathbf{n}.\mathbf{l})$) that accounts for the angle at which the light is striking the surface:

$$I_{\text{lambertian}} = c \times I_{\text{in}} \times \max(0, \mathbf{n}.\mathbf{l}) \tag{1}$$

In this model, surfaces facing away from the light appear black and flat: details about the shape are lost. The *half-lambert* shading model is a simple non-physical alteration of the lambertian model to reveal those shadowed surfaces:

$$I_{\text{half-lambert}} = c \times I_{\text{in}}(\frac{1}{2} + \frac{1}{2}\mathbf{n}.\mathbf{l}) \tag{2}$$

Note that the geometric term has been replaced by the *unclamped* geometric term $\mathbf{n}.\mathbf{l}$, and rescaled so that it lies in $[0; 1]$. This simple modification appears in many stylized shading models [Goo+98; RBD06; MFE07], as a way to convey the shape of an object more clearly, regardless of lighting conditions.

In illustration, changes in illumination of a surface are also depicted by hue shifts in addition to variations in luminance: notably, the shadows tend to be depicted with a cool color instead of black. From this observation, Gooch et al. [Goo+98] proposed a shading model that reproduces these hue shifts by using a cool-to-warm color map, obtained by combining a blue-to-yellow gradient with the lambertian shading gradient (black to object color). The half-lambert modification is used to interpolate into this color map, so that the form is revealed even on surfaces facing away from the light.

Similar hue shifts are present in watercolors: illumination in watercolor is painted with layers of pigments with varying dilution. More pigments are deposited on dark regions, while brightly lit regions actually correspond to an absence of pigments. The varying pigment density produces changes in the color temperature in addition to changes in value [LM01]. Although simulation models for the diffusion of watercolor pigments exist [Cur+97], most real-time watercolor stylization techniques use simplified shading, filtering, and color modification models to approximate diffusion effects and color variations due to pigment density [LM01; BWK05; Bou+06; MSR16].

These shading models work best when it is possible to adjust the lighting accordingly. However, in some contexts, it is not possible to control precisely the lighting. This is notably the case in video games, where designers have only limited control over the position of the lights relative to characters, as the latter can be moved around the scene. Yet, for gameplay purposes, characters should be clearly identifiable at all times, in widely varying lighting conditions. Mitchell,

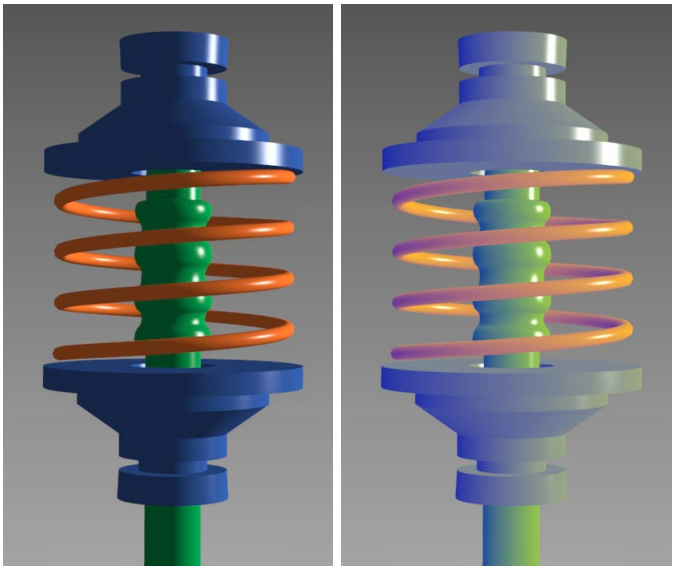


Figure 4: Comparison between standard Phong shading and Gooch shading. With Phong shading, surfaces facing away from the light appear flat, and surface detail is lost. Gooch shading reveals all surface details, even when they do not face the light. Image source: [Goo+98].

Francke, and Eng [MFE07] described the shading techniques used to effectively convey the shape and silhouettes of stylized characters in the video game *Team Fortress 2*. The shading model they described is composed of many components: an ambient term, diffuse and specular terms, and a rim-lighting term to reveal silhouettes. This shows that shading techniques can rapidly grow complex in highly dynamic environments: in our work, we seek a way to rapidly explore combinations of shading terms to design such complex models.

To properly depict surface details at various scales, many non-physical shading techniques have been proposed: Rusinkiewicz, Burns, and DeCarlo [RBD06] used a custom shading model based on the half-lambertian to enhance the perception of both the overall shape and details. This shading model is modulated by a user-controlled exaggeration parameter. By evaluating the shading at multiple scales, from fully detailed to heavily smoothed geometry, the user can separately control the degree of emphasis of the overall shape and of the surface details. Another common way to enhance the shape details is to modulate the basic shading term with *surface curvature* to better reveal ridges and creases [Ver+09; Ver+10; Ver+11]. Vergne et al. [Ver+08] proposed a generalization of curvature-based shading: through the *apparent relief* descriptor, they are able to extract specific features of a shape, such as ridges, valleys and flat regions with more flexibility and more intuitive control than previous methods. The extracted information can then be used as an additional input to stylized shading models, to increase the range of achievable appearances. Note that some of these techniques make use of screen-space filtering to enhance shading.

Toon shading The intent of toon shading techniques is to mimic light-and-shade depiction in cartoons and cel animation. They use few colors (typically under five colors) to depict illumination gradients. Visually, this results in characteristic color bands. The position and size of the bands can be adjusted by artists through a color ramp.

Many extensions to the basic toon shading model have been proposed: for instance, X-toon

[BTM06] extends the traditional toon shading by allowing users to vary tone detail according to a view-dependent scene attribute (e.g. the scene depth), effectively replacing the traditional 1D toon color ramp with a 2D texture (Figure 5). Another extension of toon shading was proposed by Todo, Anjyo, and Igarashi [TAI09]: their approach can reproduce various expressive lighting effects used in hand-drawn cartoon animation, such as the straight lighting effects used for the depiction of flat reflective surfaces.

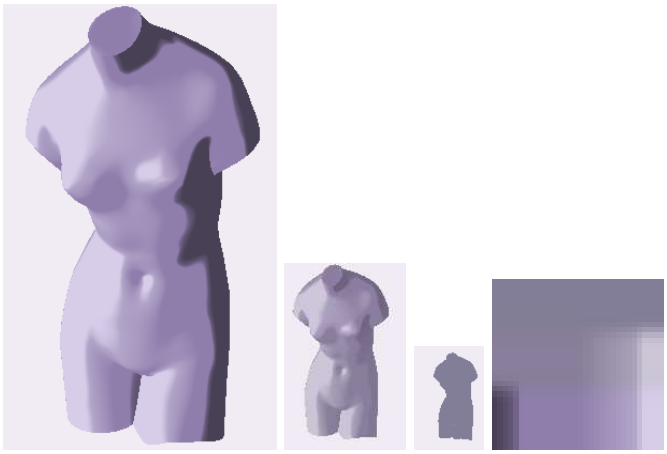


Figure 5: X-Toon shading [BTM06] example at different abstraction levels and associated 2D toon texture map. The abstraction level can be linked to view-dependent scene attributes, such as scene depth. Images taken from the paper.

Highlights Specular highlights convey important material clues about a surface. When depicting them, illustrators also take great liberties with physics. Recognizing this, several techniques have been proposed to provide artistic control over the shape and behavior of highlights on a surface. Anjyo, Wemler, and Baxter [AWB06] propose a method to alter the shape of a cartoon highlight in various ways: translation, deformation, rotation, squaring and splitting (see Figure 6). These properties can be animated over time with a keyframing system. BRDFshop [CPK06] allows one to create physically correct BRDFs from hand-painted highlights instead of indirectly manipulating numerical values. Finally, in the system of Pacanowski et al. [Pac+08], the shape and color gradient of the highlight can be directly sketched in a 2D plane oriented perpendicularly to the reflected direction.

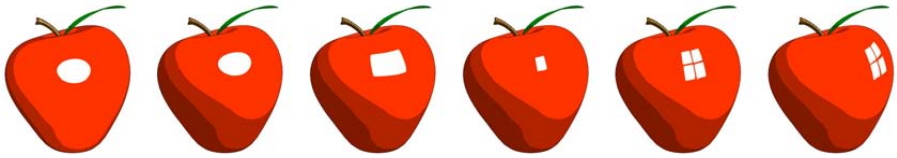


Figure 6: Tweakable light and shade: dragging the highlight on a surface, squaring, scaling, and splitting. Image source: [AWB06]

However, all these techniques are specialized for specular highlights. While they provide good artistic control and can lead to highly stylized results, they tend to have a very localized impact on the final image. In this work, we seek a more generic approach for artistic control of shading:

our intuition is that there is value in a system that would use the same kind of primitives for various shading effects, and that would allow artists to combine those primitives in novel ways, instead of considering specific effects in isolation.

This is similar to *shade trees* [Coo84], and more recent graphical shader editors [3] [4] that represent the shading equations as a tree of operation nodes. However, while these approaches are very flexible, they are also very low-level, to the point that they approach the same degree of flexibility as shader code. Thus, they can be seen as a way of structuring and visualizing shader code, but do not reduce its inherent complexity and do not immediately facilitate exploration of different appearances.

In the context of 2D vector illustration, a closer approach is *Vector Shade Trees* [Lop+13]. This system can be used to construct complex appearances by arranging a set of basic *shade nodes* in a compositing tree. These nodes are specialized vector primitives that are derived from illustration guidelines for material depiction. With their system, an artist can quickly imitate the appearance of transparent, translucent or reflective objects by combining a few of these primitives. This is close to what we want to achieve in the context of 3D scenes: one of our goals is to allow artists to use some of these guidelines in the context of 3D animated scenes with dynamic lighting environments.

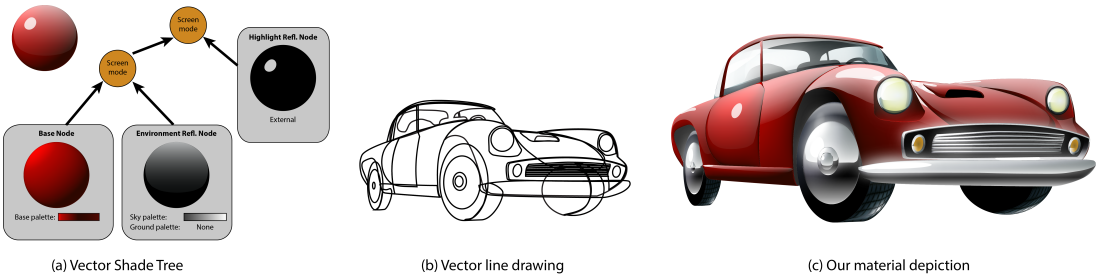


Figure 7: Material design in 2D illustration with *Vector Shade Trees*. The user arranges multiple basic shading primitives in a compositing tree (left). The material can then be applied to vector line drawings (middle, right). Image source: [Lop+13]

Our approach is most closely related to the work of Vanderhaeghe et al. [Van+11] with *Dynamic stylized shading primitives*. Their system can reproduce various stylized shadings with a layered combination of primitives. Primitives are composed of a base shading parametrization that can vary continuously between diffuse and specular behaviors, and parameters can be tweaked to control the anisotropy of the shading effect, and the shape of the intensity profile. The approach we propose is similar, but with lower-level primitives: Instead of having a single adjustable parametrization, we allow users to freely mix-and-match simpler parametrizations representing different shading behaviors (e.g. diffuse, specular, etc.) and scene properties (e.g. curvature) to create novel behaviors.

In the recent work on *barycentric shaders* by Akleman, Liu, and House [ALH16], shading is art-directed via user-provided *control images*, which are then mixed according to *weight images* derived from usual shading parametrizations (diffuse, specular, rim-lighting, depth) and remapping functions.

This is similar to the system we propose, although our system does not incorporate local control via textures, and we adopted a lower-level approach in which we chose to directly expose

[3] Unreal Engine: Essential Material Concepts `https://docs.unrealengine.com/en-US/Engine/Rendering/Materials/IntroductionToMaterials`

[4] Unity: Shader Graph `https://unity3d.com/shader-graph`

the functions that define the illumination profile as discrete 1D texture maps, editable on-the-fly by the user with various tools. Additionally, we expose basic operations on parametrizations to combine them and create new shading behaviors, and locally perturb them to convey hints about the small-scale geometry of the surface.

4 Overview

Shading design is a tedious process, especially with highly dynamic 3D scenes. We've seen that a lot of different models exist, yet they share some common terms in their formulation. For example, diffuse effects are always based upon the geometric term $\mathbf{n} \cdot \mathbf{l}$. Similarly, the surface curvature is often used in shading models for emphasizing details. These models only differ by how they map those basic shading terms to actual colors on the screen.

Designers know how to combine these terms into full models to achieve artistic goals: reveal shapes in more detail, convey mood in a scene effectively, etc. But this is often done through a tedious process of fine-tuning coefficients of shading terms and writing shader code. In this chapter, we propose a tool to facilitate exploration of, and experimentation with, shading models. In our tool, the user designs a shading by layering one or more *shading effects* each representing one independent component of the resulting shading. We structure the design process of individual shading effects in clear, distinct steps:

- Choosing and tuning the *behavior* of a shading term. This is done through combinations of *base parametrizations* and *value maps*: they control how the shading effect will move and spread on the object when changing viewpoints or lights. Base parametrizations are modeled after the basic terms common to many shading models.
- Controlling the color gradient. The illumination gradient of a shading effect is controlled through a 1D color map, directly editable by the user.
- Adding small-scale details to shading effects using *perturbations* that locally modify parametrizations. To keep the appearance spatially and temporally coherent, details are generated using 3D procedural noises.

An overview of the different steps of our workflow is provided in Figure 8. A typical session with our tool starts with the user loading the object onto which they wish to design the shading, and setting the lighting configuration (number and position of lights). From then on, users can refine the appearance of the object on the screen by adding one or more shading effects in a layered fashion. The design process is fully dynamic: at any point the user is able to move the camera or the lights and the displayed appearance will update accordingly.

From a technical standpoint, the system is implemented on the GPU as a series of screen-space operations on standard G-buffers: surface normals and tangents, depth, etc. It thus falls in the category of *deferred shading* techniques.

5 Shading behaviors

Shading behaviors describe the location and extent of a shading effect and its dynamic behavior when moving the object, the light, or the viewpoint. Users of our system design behaviors by first choosing and optionally combining *base parametrizations*. Then, a value map is applied on base parametrizations, to adjust the intensity profile of the resulting effect.

Formally, since we are in a deferred shading configuration, a shading behavior can be seen as a screen-space value: $\text{shade}(x, y)$, with (x, y) being the screen-space position. In more detail, we

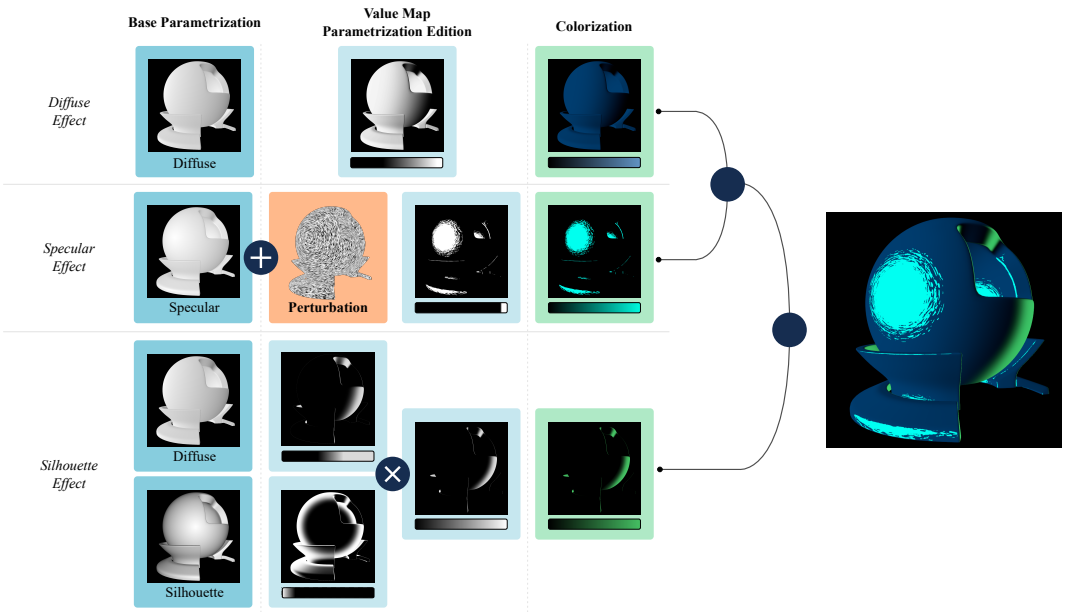


Figure 8: Illustration of our workflow showing an example with three appearance effects. A user can modify and combine *base parametrizations* to design the shading behavior (blue nodes) of an appearance effect, using value maps and combination operations. A color map (green nodes) is then applied on the designed behavior to colorize the effect. Output effects are then composited to obtain the final appearance. Perturbations (orange nodes) can be attached to every operation in order to add procedural details to an effect. The orientation of the perturbation can be controlled by the gradient of a shading behavior (as shown here), or by an external vector field, such as a tangent map.

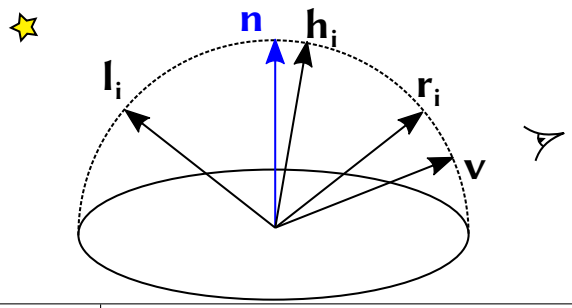
define a shading behavior as the application of a *value map* $f : [0, 1] \rightarrow [0, 1]$ on a *parametrization* $p(x, y) \in [0, 1]$. For clarity we omit (x, y) in the rest of the chapter. Thus:

$$\text{shade} = f\,(p)$$

The parametrization p is a screen-space term derived from G-buffers (normals, depth, etc.) and global scene parameters (light positions, viewpoint) and that describes a base shading behavior. We provide several *base parametrizations* that represent common shading behaviors. They can be used as-is or combined to produce more complex behaviors.

5.1 Base parametrizations

In computer graphics, shading models are composed of a sum of shading terms that model different behaviors. For instance, the Phong shading model [Pho75] has both a *diffuse* term and a *specular* term, for diffuse light and specular highlights. Individually, most shading terms depend on scalar values derived from the local geometry of the surface, the viewpoint, and the position of the lights. For example, terms that account for diffuse lighting have a dependency on the *geometric term* $\mathbf{n} \cdot \mathbf{l}$ (the cosine of the angle between the light and the surface normal); specular highlights depend on $\mathbf{n} \cdot \mathbf{h}$ or $\mathbf{r} \cdot \mathbf{v}$ depending on the model; Fresnel reflection effects are usually approximated with a term that depends on $\mathbf{n} \cdot \mathbf{v}$. In this work, we call these values



$\mathbf{n}$	Surface normal
$\mathbf{l_i}$	Light direction: unit vector to light i
$\mathbf{r_i}$	Reflected light direction
$\mathbf{v}$	View direction: unit vector to eye position
$\mathbf{h_i} = \frac{\mathbf{l_i}+\mathbf{v}}{\lvert\mathbf{l_i}+\mathbf{v}\rvert}$	Halfway vector between the light direction i and view direction

Figure 9: Local geometry used in the definition of base parametrizations.

parametrizations. They determine the global behavior of a shading term on a surface: how it will move and spread in response to light and view changes.

As in the technique of Akleman, Liu, and House [ALH16], we propose to use several base parametrizations to encode basic shading behaviors. These are derived from observation of the terms that appear frequently in shading models. In addition to terms derived from the local surface geometry used in physical models, we also add screen-space terms such as the screen-space surface curvature, which can be used to enhance shape features.

Diffuse behavior: Diffuse shading effects depend only on the geometric term $\mathbf{n}\cdot\mathbf{l}$. Notably, they are not view-dependent and do not slide on the object when moving the viewpoint. In our workflow, diffuse behaviors are parameterized by an angular remapping of the geometric term:

$$p_{\text{diffuse}} = 1 - \text{acos}\left(\mathbf{n}\cdot\mathbf{l_i}\right)/\pi$$

This limits the distortion of illumination profiles on surfaces. A similar remapping is employed in the system of Vanderhaeghe et al. [Van+11]. Similarly to the half-lambert model, the geometric term is not clamped between 0 and 1 and can be used to reveal geometry that is not facing the light. Note that there is a base diffuse parametrization for each light in the scene.

Specular behavior: Similarly, specular behaviors are parameterized by:

$$p_{\text{specular}} = 1 - \text{acos}\left(\mathbf{n}\cdot\mathbf{h_i}\right)/\pi$$

where $\mathbf{h_i} = \frac{\mathbf{l_i}+\mathbf{v}}{\lvert\mathbf{l_i}+\mathbf{v}\rvert}$. They are used to add specular highlights to objects.

Silhouette: shading behaviors that affect the object silhouettes are parameterized by:

$$p_{\text{silhouette}} = 1 - 2\times\text{acos}\left(\mathbf{n}\cdot\mathbf{v}\right)/\pi$$

This term ranges from 0 at grazing view angles, to 1 on normals oriented towards the camera. In shading models, dependencies on the $\mathbf{n}\cdot\mathbf{v}$ term typically appear in *rim-lighting*

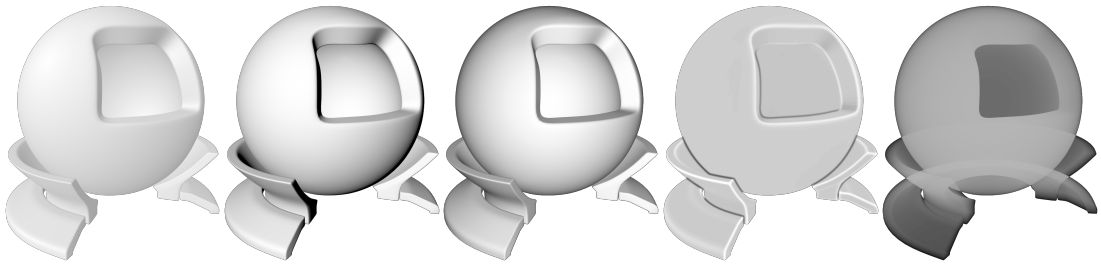


Figure 10: Visualization of base parametrizations on a given object, for a given light position and viewpoint. Parametrizations are in the $[0, 1]$ value range, from black to white. Users are able to see the how a parametrization evolves by dynamically moving lights and the camera.

terms, to position lighting effects on silhouettes, or to mimic the contribution of a Fresnel term.

Curvature: Curvature shading effects are parameterized by the screen-space view-dependent mean curvature κ [Ver+11], remapped in the $[0, 1]$ range:

$$p_{\text{curvature}} = 0.5 \times (1 + \tanh(s \times \kappa))$$

where s is a user-controllable parameter to adjust the covered value range. It can be used in combination with other parametrizations to position shading effects on sharp object features.

Thickness: the thickness parametrization is defined by the distance between the projected front and back object faces.

$$p_{\text{thickness}} = |z_{\text{front}} - z_{\text{back}}|$$

This requires a slight modification to the classic rendering pipeline to keep track of both the frontmost and the backmost depths. We found that it can provide a good visual approximation of translucency effects for simple geometries that do not have large hollow parts.

A visualization of those parametrizations on a test object for a given light position and viewpoint is provided in Figure 10.

5.2 Value maps

The shading behavior described by a parametrization can be refined with the value map f, stored in a 1D texture. By default the value map is initialized to the identity function and does not modify the behavior of the parametrization p, so the user sees one of the images of Figure 10. It can be modified it to, for instance, increase the value range of a parametrization, modify the falloff at shading terminators in a manner similar to Mitchell, Francke, and Eng [MFE07], introduce toon-like hard value transitions, or change the spread and falloff of specular highlights (see Figure 11).

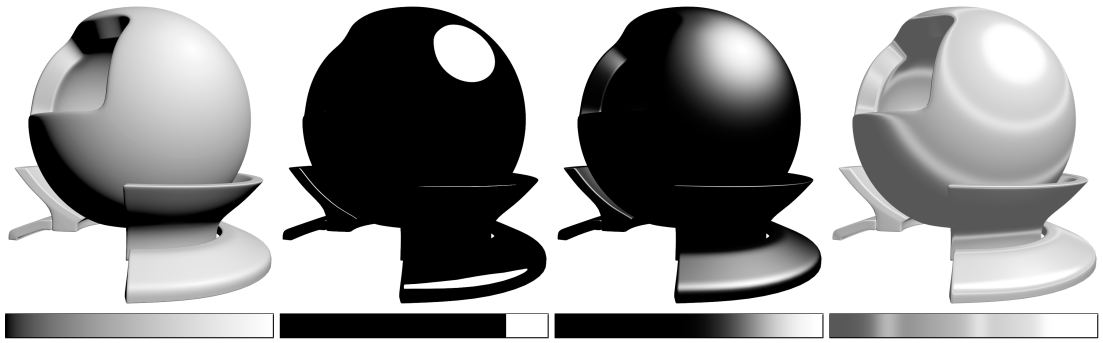


Figure 11: Use of value maps on a specular base parametrization to produce highlights of varying glossiness, and a stylized effect.

5.3 Composition

A user can choose one of the base parametrizations unmodified, or can design more complex behaviors by combining them together using traditional compositing operators: *multiply*, which can be used for masking part of parametrizations, akin to a logical *and* operation; and *screen*, which has the opposite effect of multiply and can be used to merge the behavior of two parametrizations together (logical *or*). In formal terms, given two parametrizations p_a and p_b:

$$p_{\text{screen}} = 1 - (1 - p_a) \cdot (1 - p_b)$$
$$p_{\text{multiply}} = p_a \cdot p_b$$

This compositing approach to combining parametrizations provides the most flexibility and is also more intuitive to users already familiar with compositing software. In tandem with value maps, the composition operators can be used to restrain the location and spread of a shading effect. For instance, assume that the user wants a shading effect that behaves like specular highlights, but that is only present on sharp edges. This can be achieved with a base *specular* parametrization, multiplied by the *curvature* parametrization with a value map to filter desired areas of high curvature.

6 Perturbation terms

Parametrizations can be modified with procedural noise before applying a color map, in a manner similar to domain warping. In our system, this is implemented as *perturbations*: optional operations that modulate the parametrization p of a shading behavior with a locally varying *perturbation term* d based on solid procedural noise.

Warping locally modifies the behavior of a parametrization: it allows details that are revealed by the shading effect. In constrast, compositing the details over the final result would produce details everywhere regardless of shading and would not affect the underlying behavior of the shading. Furthermore, procedural noise allows users to quickly experiment with adding visual details on an object, without needing to modify the geometry or to manually paint textures.

In the following paragraphs, p' is the perturbed parametrization, p is the original parametrization p' is the combination of the original parameter p with a spatially varying *perturbation term* $d \in [0, 1]$, given by the evaluation of a procedural noise, using one of the operations below:

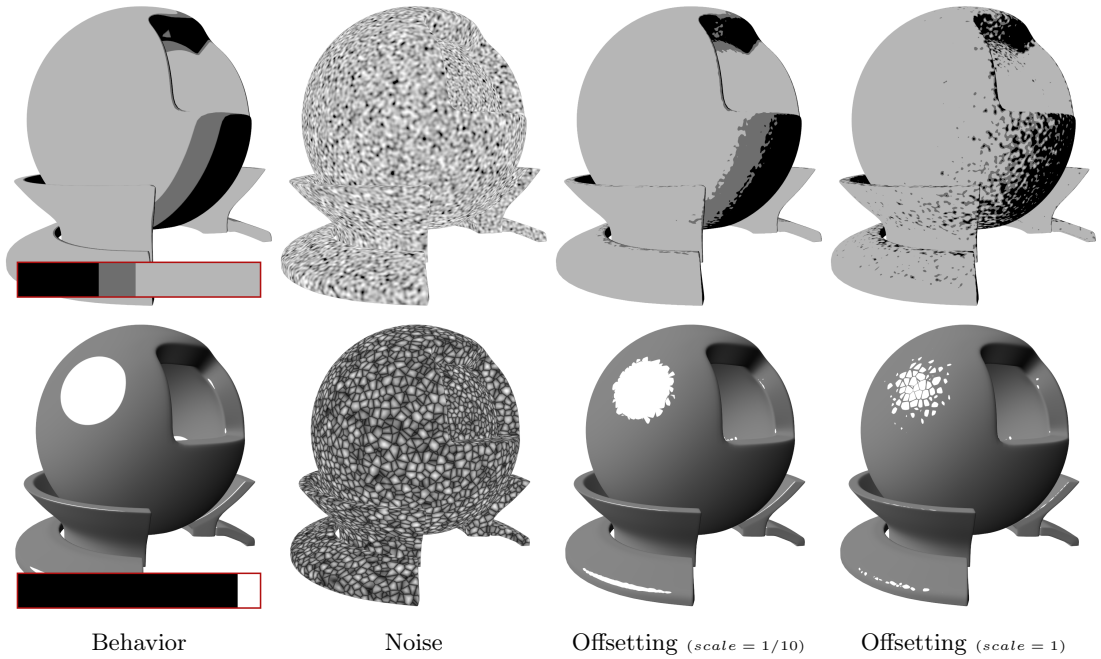


Figure 12: Offsetting examples. (Top) A toon shading transition offset with a gradient noise. (Bottom) A specular behavior offset with a cellular noise produces highlights with complex shapes. A high scale parameter increases the size of the jitter effect.

Offsetting p' is given by offsetting the original parameter with the perturbation term and clamping the result in $[0, 1]$, following the formula

$$p' = \text{clamp}\,(p + scale \times (d - 0.5))$$

where *scale* is a user-defined scaling factor. A typical use for this operator is to jitter the boundaries of *hard* shading features introduced in a value map (such as terminators or hard specular highlights). The strength of this effect is controlled by the scaling factor *scale*, as shown in Figure 12. For parametrizations that depend on the surface normal, the offsetting operation can reproduce effects that are perceptually similar to bump mapping.

Texturing p' is given by:

$$p' = p \cdot d$$

The perturbation term will affect the object globally instead of being localized at shading terminators. Thus, in contrast with offsetting, this operation can be said to add textural details. A usage example of this operator is given in Figure 15(g).

Multiple perturbation steps can be chained together allowing complex dynamic stylized appearances. The perturbation term d comes from the evaluation of a 3D (solid) procedural noise in model space. This avoids the shower-door artifacts commonly found with 2D procedural noise. Another advantage of procedural noises compared to bitmap images is that they do not need any surface parametrization and can be evaluated in real-time on animated, deformable objects. In our deferred implementation, the solid noise is evaluated in screen-space, using the G-buffer containing the 3D model-space position. This results in a 2D image containing perturbation

values at all positions on the screen. In our implementation, we provide two choices of noise models to generate this perturbation term:

Gradient noise: A grayscale solid gradient noise (Perlin noise) [Per85]. The user can control its frequency, amplitude (centered on 0.5) and number of octaves. For example, low amplitude gradient noise used with offsetting can produce a bumpy appearance on objects with hard shading or sharp highlights, as show in Figure 12.

Cellular noise: This is the model described by Worley [Wor96]. It generates structured details resembling scales with a user-defined frequency and regularity. Figure 12 shows cellular noise perturbing a specular parametrization to alter the shape of a specular highlight and give the impression of surface irregularities.

6.1 Line Integral Convolution

Additionally, to increase the range of achievable appearances with procedural solid noise, a user can optionally apply a *Line Integral Convolution*(LIC) [CL93] filter on the generated solid noise. LIC filters are mostly used for the visualization of vector fields, but have also been used for stylization purposes [LM01]. The technique we propose in our pipeline is similar to the latter: the apply the LIC filter in the 2D image containing the generated solid noise. Two options are proposed to the user for the vector field used to guide the LIC filter:

1. The LIC is guided by the screen-space gradient of the parametrization being perturbed. This gradient can optionally be rotated 90 degrees. This allows brush- and sketch-like effects that are oriented according to shading features.
2. The LIC is guided by a vector field defined on the surface of the object, projected in screen-space (for instance, the projected surface tangents or normals).

We show in Section 9 that this step allows the user to approximate a range of anisotropic shading effects and appearances, such as a brushed metal look, view- and light-dependent cross-hatching effects, or painterly strokes oriented according to the illumination falloff.

7 Colorization and compositing

Finally, the colorized result of one appearance effect is obtained by applying a 1D color map on the shading behavior: a 1D color map is a function $f : [0, 1] \rightarrow [0, 1]^4$ stored in a 1D RGBA texture. An artist can edit this color map to change the shading tones, alpha channel and falloff of shading features. Note that there is only one color mapping operation per shading effect.

The result of individual shading effects are then blended together according to a stack of compositing operations to obtain the final appearance, as shown in Figure 8. In addition to regular alpha blending, several standard compositing operations are available at this stage, including the *multiply*, *screen* and *overlay* blend modes.

8 User interface

During the design process, the user has to edit 1D value maps and color maps. We chose to expose them directly instead of having a model with parameters for artistic flexibility. However, editing these maps can also be tedious: to alleviate this, we provide editing tools to facilitate

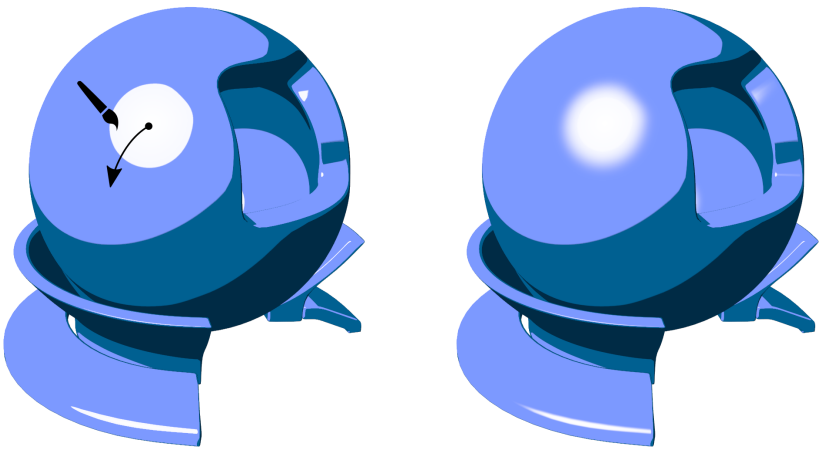


Figure 13: Blurring a specular highlight with the click-and-drag interface on the object. (Left) With the highlight color map currently selected, the user draws a stroke on the object to define the range of parameters to blur on the color map. (Right) Result of the blur operation.

frequent operations done on color and value maps, which should be familiar to users of digital painting software:

Flat The *flat* tool paints a constant color or value over the selected parameter range. This is useful to create toon-like color bands.

Gradient The *gradient* tool paints a linear gradient. It can be used to paint smooth tonal variations.

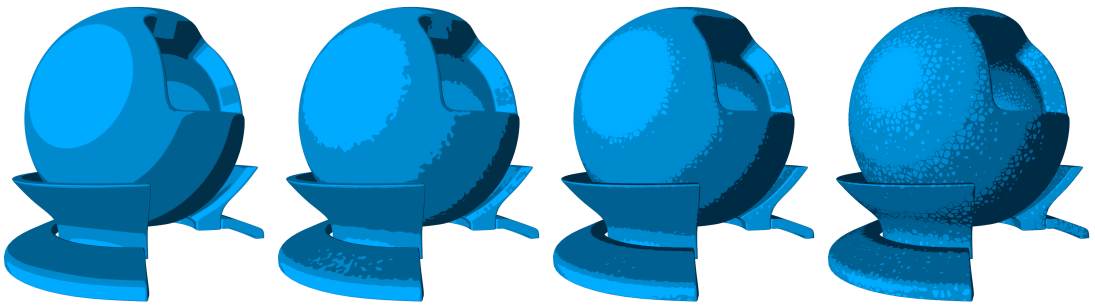
Blur The *blur* tool applies a 1D gaussian blur over the selected parameter range. A typical use case is to increase perceived glossiness of an object by blurring the falloff of a specular highlight.

These tools operate on a parameter range of the map. The range can be selected by clicking and dragging 1D view of the map, or directly on the object by drawing a stroke: in the latter case, the parameter range is determined by looking up the values of the associated parametrization at the endpoints of the stroke. An example of user interaction is shown in Figure 13.

9 Results

Our workflow is implemented in C++ and OpenGL. All operations are done on the GPU. While there is considerable room for improvement on the performance of our implementation (most of our image passes could be fused in one single shader pass), our system still keeps interactive frame rates ($>$ 20 FPS), even for complex styles. Note that our system only requires normal and depth maps for calculating the base parametrizations, and position maps for coherent 3D noise. Thus, it can be easily integrated into any pipeline that produces these three outputs.

A user starts by loading a mesh from a file, and by adding a first appearance effect. The initial view shows the default shading behavior. From then on, the user can edit the value map or color map by clicking and dragging on the mesh or directly on the associated 1D textures, add a perturbation operation, or add new appearance effects. All modifications are reflected



(a) (b) (c) (d)

Figure 14: Simple toon shading (a). Our system easily allows details to be added with different offsetting noises (b,c) and offsetting scales (c,d).

in real-time on the mesh. The designed appearance stays independent of the model used for editing: this allows the user to change the mesh during a design session and resume editing operations on this new mesh seamlessly. Our interface also provides controls for adjusting global parameters, such as the position of the lights (azimuth and elevation), or the scaling parameter for the surface curvature. These controls are shared by all appearance effects. The user can also move the camera at any time by clicking and dragging on the object. A example of editing session is provided in an accompanying video [5].

Directional lights can be added in a scene as needed, each light having its own associated *Diffuse* and *Specular* parametrizations. This way, a user can bind shading features to one particular light. Their directions can be controlled individually, or rotated all at once as a single lighting environment. By progressively adding different effects that depend on different lights, users can emulate complex lighting environments, as shown in Figure 15(e).

Figure 14 shows how our framework can accommodate simple toon shading effects with one appearance effect parameterized by a *Diffuse* behavior (a). In (b-d) we add a perturbation operation with the offsetting mode, using unmodified gradient noise (b), and cellular noise (c,d). In (c,d) we vary the scale parameter s of the offsetting to increase or decrease the amount of added detail.

In Figure 15, we show various styles achievable with our system. Simple toon shading (a) is implemented with only one appearance effect. The shading tones were painted directly on the object using the flat tool. In (b), the same color map was smoothed and parameterized by a *Silhouette* behavior instead of *Diffuse*. A glossy specular highlight was added as a second appearance effect. Figure 15(c) illustrates the use of the thickness base parametrization to give a convincing impression of a translucent material. In (d), we combined a simple diffuse effect with a silhouette lighting effect appearing only in unlit regions. In (e), we illustrate the ability of our system to bind appearance effects to different lights: two glossy specular effects were added and parameterized by two different lights. In (f), we offset a specular highlight with a high-frequency cellular noise to reproduce a glinting effect. The use of procedural noise allows a coherent result when moving the light. In (g), we illustrate the effect of the *texturing* perturbation mode. Two perpendicularly oriented noises are combined with a diffuse behavior using a texturing operation. The result is oriented along the image-space gradient of the diffuse term and follows its variations. Finally, in (h) we re-used the metallic appearance shown in (b) and applied an offsetting operation with an oriented noise in order to reproduce a brushed metal appearance. Some of those styles

[5] `https://vimeo.com/289635367`

can be seen under dynamic viewpoints and lights in the accompanying gallery video [6].

In Figure 16 we illustrate the behavior of a complex combination of appearance effects under varying lighting conditions. The appearance shown is composed of a base diffuse effect (orange), and an edge lighting effect (cyan) that appears only on object silhouettes and regions of high curvature. The behavior of this effect was designed by merging the *Silhouette* and *Curvature* behaviors using the *Screen* mode, and by multiplying the result with a remapped *Diffuse* behavior, so that the effect appears only in the unlit part of the object. By providing simple compositing operations to combine and remap shading behaviors, our system allows fast prototyping of such complex shading behaviors without the need for extensive technical knowledge.

Figure 17 shows the decomposition of an appearance effect based on surface curvature: we combined, using the multiply blending mode, value-mapped *Specular* (b) and *Curvature* (a) behaviors to obtain highlights localized on sharp object edges that behave specularly (c). The final result is shown in (d). We then transferred this style to other objects, as shown in the bottom row. The size of the edge highlight effect can be modified either by tweaking the value map used to select the desired curvature range or by adjusting the global scaling parameter for curvature.

10 Discussion

We proposed a workflow and a set of editing tools to design shading models that produce stylized appearances that stay coherent under light and viewpoint changes. Our main contribution is the separation of individual effects in three aspects: *shading behavior* of the effect, addition of *procedural details*, and *colorization*. This decomposition offers a more precise and predictable result than offline appearance transfer techniques, while being more flexible and easier to use for non-technical artists than specialized shading models. Our workflow allows stylization results to be calculated in real-time and is thus usable in interactive contexts such as visualization, 3D modeling, or video games. We have provided several base parametrizations that encode common shading behaviors and proposed ways to control their size and placement on an object. For non-technical artists, we feel that this approach is more direct for designing complex behaviors than hand-written shaders.

10.1 Parametrizations

We feel that the proposed system could benefit from more base parametrizations: notably, a current limitation is the inability to reproduce convincing refraction effects for translucent objects. More work is needed in this direction to provide intuitive base parametrizations for these effects. The same is true for global illumination effects such as color bleeding between objects in a scene, or subsurface scattering, for which we should provide specialized parametrizations.

Currently, the parametrizations are only defined on pixels covered by the rasterized object. This means that all shading and appearance effects are cut at object silhouettes. A possible improvement to our system would be to extend parametrizations in a region surrounding the object in order to create appearance effects that alter the object silhouette, such as a glow effect.

Concerning procedural details, the combination of 3D gradient noise and line integral convolution ended up being sufficient for a wide range of appearances. Our system can also be used with cellular noise to produce medium-scale details, but could be extended with other kinds of structured noises. The use of 3D noise produces details that are fully coherent with scene motion. However, there are cases where a more 2D aspect is desired: in this case, our stylization

[6] `https://vimeo.com/289635608`

approach could be extended with coherent 2D noise primitives [Bén+10]. Another limitation of our system in this aspect is that small-scale procedural details on shading features are not preserved when transferring the appearance to a mesh with high-frequency geometric detail. In these scenarios, we would want shading effects to ignore the high-frequency detail. This could be done either by rendering parametrizations with a smoothed out mesh, or by prefiltering the existing parametrizations to remove high-frequency variations. For this, inspiration could be taken from the *exaggerated shading* technique [RBD06], which operates at multiple detail scales.

More generally, this also raises the question of the ability of our system to reproduce shading styles found in 2D illustration. The proposed parametrizations are based on standard shading terms (e.g. the lambertian term, or the specular term), but illustrative styles may not closely follow these models. For instance, in the so-called *sculptural lighting* style described by Hogarth, each continuous part of a shape is shaded "as if the light falls on the center" (of the shape) [Hog91], independently of whether the shape is actually facing a light, or the camera: this allows revealing each continuous part of a shape with maximum clarity (Figure 18). Currently, there is no way to capture the "center" of a shape with our parametrizations: this belongs to a class of shading effects that are best defined in screen-space.

An interesting area of research would be to propose new view-depdendent shading terms beyond lambertian and specular, that include screen-space values (for instance, the screen-space distance to contours), which would possibly allow a formalization of such illustrative shading styles.

10.2 User study

As a future work, we would like to conduct a user study on artists, and measure the time saved compared to writing shader code by hand. A possible evaluation protocol would be to have artists reproduce a given shading style in a limited time using both approaches (directly writing shaders, and using our tool) and compare the time spent for each approach. It would also be interesting to compare our layered approach to shading design with the node-based shader editors implemented in many game engines and 3D modeling software. This would allow us to compare the efficiency of a representation in successive layers versus a more free-form node graph representation. With a similar protocol, we would also like to evaluate the ability of our system to reproduce shading styles found in 2D illustration.

It would also be interesting to perform a user study on non-technical users, outside the audience that we originally targeted. We believe, after having used the system for a while, that such a tool has potential uses for non-technical users, as an easily accessible appearance design tool. Such a study would provide clues as to what kind of interface and vocabulary would be necessary for less technically-skilled users. We expect that some work will have to be done on the shading primitives exposed to the users, and also the vocabulary employed: in this work, we focused on artists who have knowledge of shading terms employed in computer graphics (e.g. diffuse and specular components, rim-lighting terms, etc.), whereas non-technical users may not be familiar with this vocabulary.

10.3 Integration into existing modeling software

One way of having more users trying our tool is to integrate it into existing 3D creation toolsets. Commercial plugins for stylized shading design exist [7] [8] [9] [10] but usually target specific looks

[7] PSOFT Pencil+ `https://www.psoft.co.jp/en/product/pencil/3dsmax/`

[8] ToonKit `http://cogumelosoftworks.com/index.php/toonkit/`

[9] Maneki® `http://maneki.sh/`

[10] cebas finalToon™ 4.0 `https://www.cebas.com/index.php?pid=productinfo&prd_id=192`

(toon shading, in particular). Our system could be easily implemented into existing rendering pipelines as a deferred shading pass. Also, while our system is currently implemented with one screen-space pass per layer for historical reasons, it could be easily merged into a single pass for effects that do not depend on screen-space filtering: thus, as a future extension, it should be feasible to export the layers as one fragment shader that would be usable in game engines, for instance.

10.4 Inference from hand-painted input

Another axis of research that we would like to explore is the possibility of automatically inferring a shading model from hand-painted inputs, in complement of the manual edition of layers. With stroke annotations painted by the user, the system would deduce on-the-fly the combination of input parametrizations that best fits the shading intent of the user.

In the general case, given a finished painting, this is difficult to do. An online approach is more suited to this case, as the user could progressively refine the result of the inference by painting additional strokes. Additional hints may be provided by the user besides strokes, e.g. in the form of constraints for the inference algorithm.

This would be similar to the style transfer method proposed by Fišer et al. [Fiš+16], although in their system, the relation between the real illumination of the scene and the painted depiction is captured globally on the whole image, and then synthesized with image analogies. In our case, the inference result would be an explicit shading model (i.e. a function of the local surface properties, viewpoint, and lights) and usable in real-time contexts. Such an approach would be especially powerful when combined with a fallback to the manual editing of layers, and possibly, with local control of the shading (e.g. through normal maps, or through perturbations): for instance, we could propose a painting interface that would let the user paint local tone variations on the surface of an object and automatically translate them into coherent local adjustments of the geometry or the normal map. This way, with additional local control, we could potentially cover both ends of the interaction spectrum, from low-level control to higher-level, global behaviors. This would be a significant step towards a comprehensive framework for stylized depiction of light and shade in 3D scenes.

References

[ALH16] Ergun Akleman, S Liu, and Donald House. “Barycentric Shaders: Art Directed Shading Using Control Images”. In: *Proceedings of Expressive 2016*. 2016.

[AWB06] Ken-ichi Anjyo, Shuhei Wemler, and William Baxter. “Tweakable Light and Shade for Cartoon Animation”. In: *Proceedings of the 4th International Symposium on Non-photorealistic Animation and Rendering*. New York, NY, USA: ACM, 2006, pp. 133–139.

[Bén+10] Pierre Bénard, Ares Lagae, Peter Vangorp, Sylvain Lefebvre, George Drettakis, and Joëlle Thollot. “A Dynamic Noise Primitive for Coherent Stylization”. In: *Computer Graphics Forum (Proceedings of the Eurographics Symposium on Rendering 2010)* 29.4 (June 2010), pp. 1497–1506.

[Bou+06] Adrien Bousseau, Matthew Kaplan, Joëlle Thollot, and François X Sillion. “Interactive watercolor rendering with temporal coherence and abstraction”. In: *International Symposium on Non-Photorealistic Animation and Rendering (NPAR)*. Annecy, France: ACM, 2006.

[BTM06] Pascal Barla, Joëlle Thollot, and Lee Markosian. "X-Toon: An extended toon shader". In: *International Symposium on Non-Photorealistic Animation and Rendering (NPAR'06)*. Ed. by Douglas DeCarlo and Lee Markosian. Annecy, France: ACM, June 2006.

[BWK05] J Burgess, G Wyvill, and S A King. "A system for real-time watercolour rendering". In: *International 2005 Computer Graphics*. June 2005, pp. 234–240.

[CL93] Brian Cabral and Leith Casey Leedom. "Imaging Vector Fields Using Line Integral Convolution". In: *Proceedings of the 20th Annual Conference on Computer Graphics and Interactive Techniques*. New York, NY, USA: ACM, 1993, pp. 263–270.

[Coo84] Robert L. Cook. "Shade Trees". In: *Proceedings of the 11th Annual Conference on Computer Graphics and Interactive Techniques*. SIGGRAPH '84. New York, NY, USA: ACM, 1984, pp. 223–231.

[CPK06] Mark Colbert, Sumanta Pattanaik, and Jaroslav Krivanek. "BRDF-Shop: Creating physically correct bidirectional reflectance distribution functions". In: *IEEE Computer Graphics and Applications* 26.1 (2006), pp. 30–36.

[Cur+97] Cassidy J. Curtis, Sean E. Anderson, Joshua E. Seims, Kurt W. Fleischer, and David H. Salesin. "Computer-generated watercolor". In: *Proceedings of the 24th annual conference on Computer graphics and interactive techniques - SIGGRAPH '97* (1997).

[Fiš+16] Jakub Fišer, Ond\vrej Jamriška, Michal Lukáč, Eli Shechtman, Paul Asente, Jingwan Lu, and Daniel Sýkora. "StyLit: Illumination-Guided Example-Based Stylization of 3D Renderings". In: *ACM Transactions on Graphics* 35.4 (2016).

[Goo+98] Amy Gooch, Bruce Gooch, Peter Shirley, and Elaine Cohen. "A Non-photorealistic Lighting Model for Automatic Technical Illustration". In: *Proceedings of the 25th Annual Conference on Computer Graphics and Interactive Techniques*. New York, NY, USA: ACM, 1998, pp. 447–452.

[Hog91] Burne Hogarth. *Dynamic Light and Shade*. Watson-Guptill, 1991.

[KPD10] William B Kerr, Fabio Pellacini, and Jonathan D Denning. "BendyLights: Artistic Control of Direct Illumination by Curving Light Rays". In: *Computer Graphics Forum* (2010).

[LM01] E B Lum and Kwan-Liu Ma. "Non-photorealistic rendering using watercolor inspired textures and illumination". In: *Proceedings Ninth Pacific Conference on Computer Graphics and Applications. Pacific Graphics 2001*. 2001, pp. 322–330.

[Lop+13] Jorge Lopez-Moreno, Stefan Popov, Adrien Bousseau, Maneesh Agrawala, and George Drettakis. "Depicting Stylized Materials with Vector Shade Trees". In: *ACM Transactions on Graphics (SIGGRAPH Conference Proceedings)* 32.4 (2013).

[MFE07] Jason Mitchell, Moby Francke, and Dhabih Eng. "Illustrative rendering in team fortress 2". In: *Proceedings of the 5th international symposium on Non-photorealistic animation and rendering*. ACM, 2007, pp. 71–76.

[MSR16] Santiago E Montesdeoca, Hock-Soon Seah, and Hans-Martin Rall. "Art-directed Watercolor Rendered Animation". In: *Non-Photorealistic Animation and Rendering*. Ed. by Pierre Bénard and Holger Winnemöller. The Eurographics Association, 2016.

[Oka+07] Makoto Okabe, Yasuyuki Matsushita, Li Shen, and Takeo Igarashi. "Illumination brush: Interactive design of all-frequency lighting". In: *Computer Graphics and Applications, 2007. PG'07. 15th Pacific Conference on*. IEEE, 2007, pp. 171–180.

[Pac+08] Romain Pacanowski, Xavier Granier, Christophe Schlick, and Pierre Poulin. "Sketch and paint-based interface for highlight modeling". In: *Eurographics Workshop on Sketch-Based Interfaces and Modeling*. 2008, pp. 7–23.

[Pel+07] Fabio Pellacini, Frank Battaglia, Keith Morley, and Adam Finkelstein. "Lighting with Paint". In: *ACM Transactions on Graphics* 26.2 (June 2007), Article 9.

[Pel10] Fabio Pellacini. "envyLight: An Interface for Editing Natural Illumination". In: *ACM Trans. Graph.* 29.4 (July 2010), 34:1–34:8.

[Per85] Ken Perlin. "An Image Synthesizer". In: *Proceedings of the 12th Annual Conference on Computer Graphics and Interactive Techniques*. SIGGRAPH '85. New York, NY, USA: ACM, 1985, pp. 287–296.

[Pho75] Bui Tuong Phong. "Illumination for Computer Generated Pictures". In: *Commun. ACM* 18.6 (June 1975), pp. 311–317.

[RBD06] Szymon Rusinkiewicz, Michael Burns, and Doug DeCarlo. "Exaggerated Shading for Depicting Shape and Detail". In: *ACM Transactions on Graphics, Proceedings of ACM SIGGRAPH 2006 (Boston, MA, July 30–August 3, 2006)* 25.3 (2006), pp. 1199–1205.

[Rit+09] Tobias Ritschel, Makoto Okabe, Thorsten Thormählen, and Hans-Peter Seidel. "Interactive Reflection Editing". In: *ACM Trans. Graph. (Proc. SIGGRAPH Asia 2009)* 28.5 (2009).

[Rit+10] Tobias Ritschel, Thorsten Thormählen, Carsten Dachsbacher, Jan Kautz, and Hans-Peter Seidel. "Interactive On-surface Signal Deformation". In: *ACM Trans. Graph.* 29.4 (July 2010), 36:1–36:8.

[Sch+13] Thorsten-Walther Schmidt, Jan Novak, Johannes Meng, Anton S Kaplanyan, Tim Reiner, Derek Nowrouzezahrai, and Carsten Dachsbacher. "Path-Space Manipulation of Physically-Based Light Transport". In: *ACM Transactions on Graphics (Proceedings of ACM SIGGRAPH 2013)* 32.4 (Aug. 2013).

[Slo+01] Peter-Pike J Sloan, William Martin, Amy Gooch, and Bruce Gooch. "The Lit Sphere: A Model for Capturing NPR Shading from Art". In: *Proceedings of Graphics Interface 2001*. Toronto, Ont., Canada, Canada: Canadian Information Processing Society, 2001, pp. 143–150.

[TAI09] Hideki Todo, Ken Anjyo, and Takeo Igarashi. "Stylized lighting for cartoon shader". In: *Computer Animation and Virtual Worlds* 20.2-3 (2009), pp. 143–152.

[TAY13] Hideki Todo, Ken Anjyo, and Shun'Ichi Yokoyama. "Lit-Sphere Extension for Artistic Rendering". In: *Vis. Comput.* 29.6-8 (June 2013), pp. 473–480.

[Van+11] David Vanderhaeghe, Romain Vergne, Pascal Barla, and William Baxter. "Dynamic Stylized Shading Primitives". In: *NPAR '11:Proceedings of the 8th International Symposium on Non-Photorealistic Animation and Rendering*. Vancouver, Canada, Aug. 2011, pp. 99–104.

[Ver+08] Romain Vergne, Pascal Barla, Xavier Granier, and Christophe Schlick. "Apparent Relief: A Shape Descriptor for Stylized Shading". In: *Proceedings of the Sixth International Symposium on Non-Photorealistic Animation and Rendering (NPAR 2008, June 9–11, 2008, Annecy, France)* (2008).

[Ver+09] Romain Vergne, Romain Pacanowski, Pascal Barla, Xavier Granier, and Christophe Schlick. "Light Warping for Enhanced Surface Depiction". In: *ACM Transaction on Graphics (Proceedings of SIGGRAPH 2009)* 28.3 (2009).

[Ver+10] Romain Vergne, Romain Pacanowski, Pascal Barla, Xavier Granier, and Christophe Schlick. "Radiance Scaling for Versatile Surface Enhancement". In: *I3D '10: Proc. symposium on Interactive 3D graphics and games*. Boston, United States: ACM, Feb. 2010.

[Ver+11] Romain Vergne, Romain Pacanowski, Pascal Barla, Xavier Granier, and Christophe Schlick. "Improving Shape Depiction under Arbitrary Rendering". In: *IEEE Transactions on Visualization and Computer Graphics* 17.8 (June 2011), pp. 1071–1081.

[Wor96] Steven Worley. "A Cellular Texture Basis Function". In: *Proceedings of the 23rd Annual Conference on Computer Graphics and Interactive Techniques*. New York, NY, USA: ACM, 1996, pp. 291–294.

[Zub+15] Carlos J Zubiaga, Adolfo Muñoz, Laurent Belcour, Carles Bosch, and Pascal Barla. "MatCap Decomposition for Dynamic Appearance Manipulation". In: *Eurographics Symposium on Rendering 2015*. Darmstadt, Germany, June 2015.

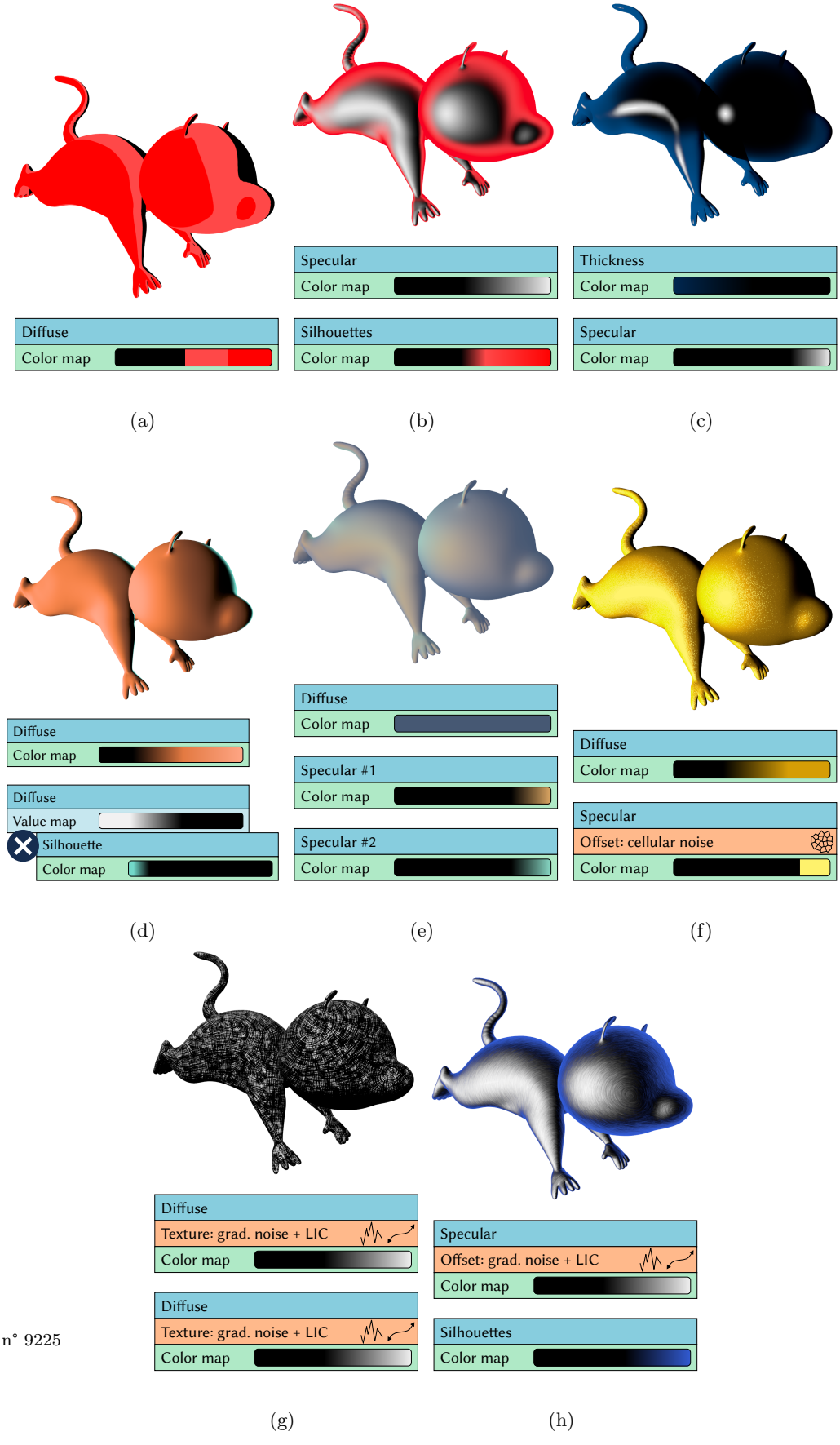


Figure 15: Various styles obtained with our system, and corresponding layer decompositions. See Section 9 for details.

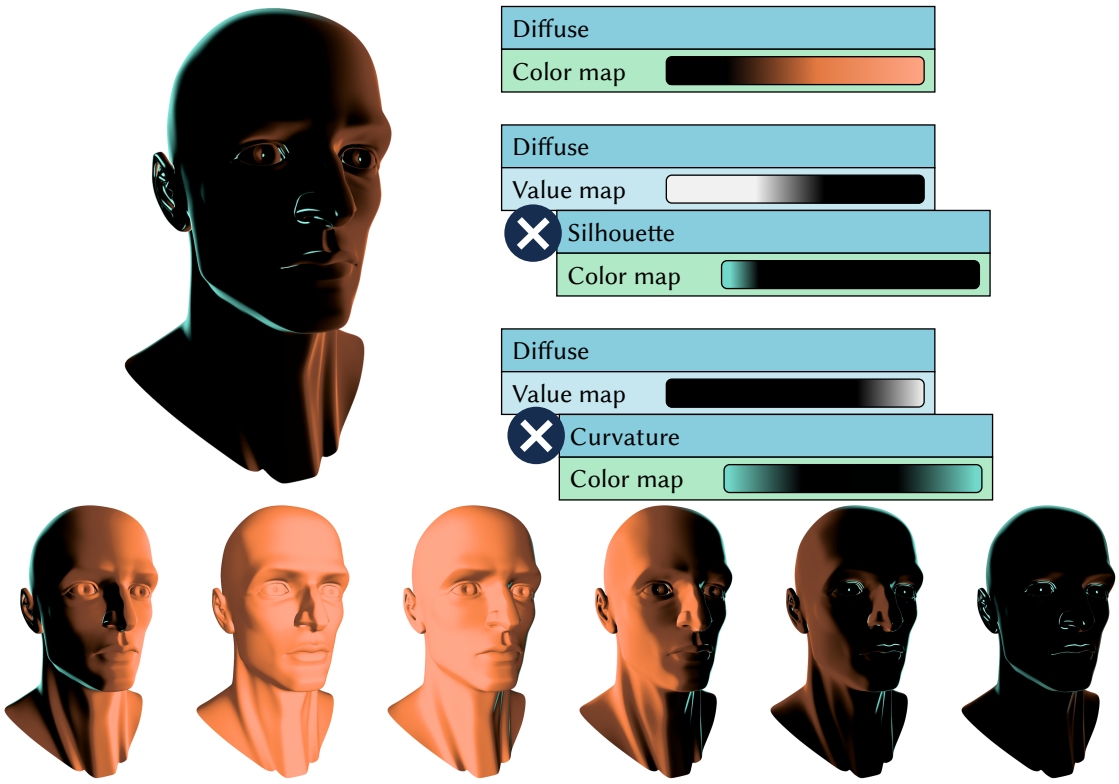


Figure 16: Our system allows complex spatially and temporally coherent behaviors with varying light directions. Here, curved regions and silhouettes are enhanced with a bluish color when the diffuse component gets darker.

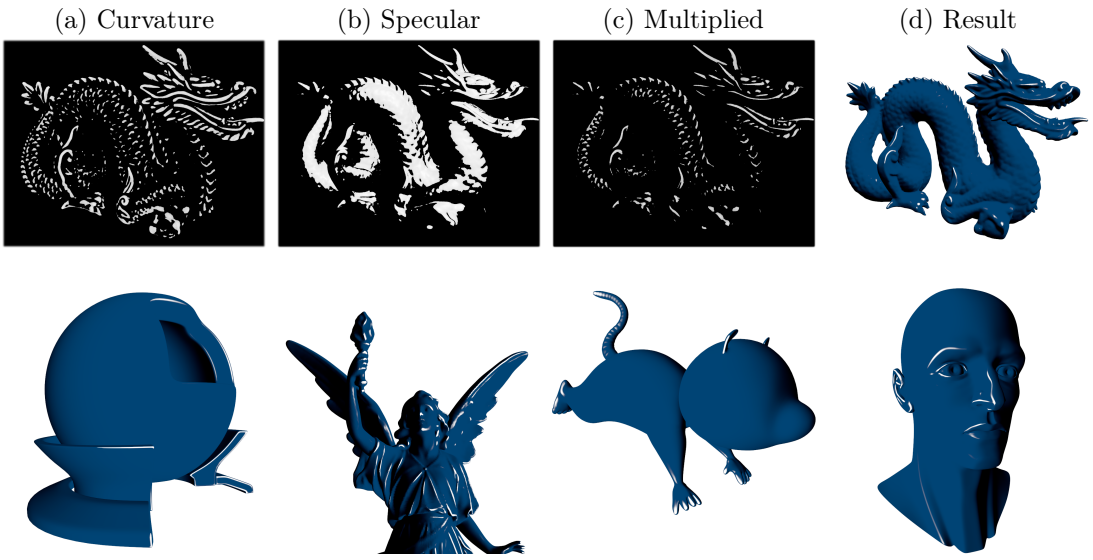
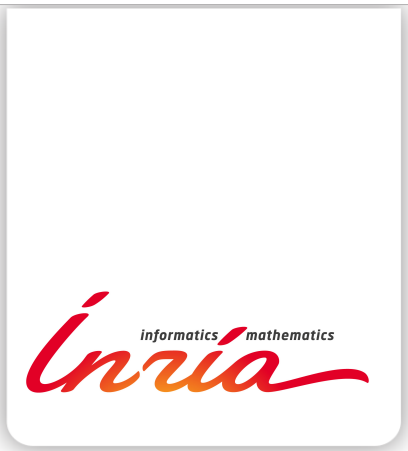


Figure 17: Combining parameterizations: user-defined curvature (a) and specular (b) behaviors are multiplied together (c) in order to ensure the highlights to appear on highly curved regions only (d). The corresponding style is then directly transferred to other input objects, as shown in the bottom row.

Figure 18: Sculptural shading example by Burne Hogarth. This shading style reveals all parts of the shape, producing highlights centered on each continuous component of the shape. Image source: [Hog91]



RESEARCH CENTRE
GRENOBLE – RHÔNE-ALPES

Inovallée
655 avenue de l'Europe Montbonnot
38334 Saint Ismier Cedex

Publisher
Inria
Domaine de Voluceau - Rocquencourt
BP 105 - 78153 Le Chesnay Cedex
inria.fr

ISSN 0249-6399